Research Article

Exploration for Countering the Episodic Memory

Rong Zhou,[1] Yuan Wang ,[2] Xiwen Zhang,[3] and Chao Wang[1]

[1]*Mechanical Engineering School, Southeast University, Nanjing, China*
[2]*Aviation Engineering School, Air Force Engineering University, Xi'an, China*
[3]*Information and Navigation College, Air Force Engineering University, Xi'an, China*

Correspondence should be addressed to Yuan Wang; wangy_af@163.com

Received 17 December 2021; Accepted 17 February 2022

Academic Editor: Ahmed Mostafa Khalil

Reinforcement learning is a prominent computational approach for goal-directed learning and decision making, and exploration plays an important role in improving the agent's performance in reinforcement learning. In low-dimensional Markov decision processes, table reinforcement learning incorporated within count-based exploration works well for states of the Markov decision processes that can be easily exhausted. It is generally accepted that count-based exploration strategies turn inefficient when applied to high-dimensional Markov decision processes (generally high-dimensional state spaces, continuous action spaces, or both) since most states occur only once in deep reinforcement learning. Exploration methods widely applied in deep reinforcement learning rely on heuristic intrinsic motivation to explore unseen states or unreached parts of one state. The episodic memory module simulates the performance of hippocampus in human brain. This is exactly the memory of past experience. It seems logical to use episodic memory to count the situations encountered. Therefore, we use the contextual memory module to remember the states that the agent has encountered, as a count of states, and the purpose of exploration is to reduce the probability of encountering these states again. The purpose of exploration is to counter the episodic memory. In this article, we try to take advantage of the episodic memory module to estimate the number of states experienced, so as to counter the episodic memory. We conducted experiments on the OpenAI platform and found that counting accuracy of state is higher than that of the CTS model. At the same time, this method is used in high-dimensional object detection and tracking, also achieving good results.

1. Introduction

Reinforcement learning, widely used in terms of the optimal control of Markov decision processes (MDPs) such as games and robotics, is a prominent computational approach for goal-directed learning and decision making [1].

The agent in reinforcement learning interacts with an environment, rather than just accepts a supervised signal, learning to map situations of a trajectory (or for some situations an episode) to actions to achieve a maximum expected cumulative reward.

When meeting high-dimensional situations, approximate solution methods were required. Deep Q-network (DQN) [2, 3] first attempted to apply reinforcement learning to high-dimensional problems by combining Q-learning with deep convolutional neural networks (CNN) as parameterized function approximators. However, this gives rise to more uncertainty of the reinforcement learning process.

All the time, in both low and high dimensions, tradeoff between exploration and exploitation is a great challenge arising in reinforcement learning for the agents interacting with unknown environments [1].

Hyperopic exploration, a main challenge in reinforcement learning, is essential for the extensively used gradient based reinforcement learning algorithms being sensitive to the initial policy, flat or deceptive gradient, and also for the uncertainty of action values estimation, to make sure that the agent is not trapped into local optimization.

In terms of the difficulty of exploration, [4] roughly classified Atari 2600 games in Arcade Learning Environment (ALE) into three taxa: easy exploration (this taxon can be divided into two according to the final scores), hard exploration with dense reward, and hard exploration with sparse reward. It is obvious that, for the above traps, different exploration strategies work. Just few strategies such as pseudocount-based exploration [4] are valid for the

well-known hard exploration Montezuma's Revenge, while most strategies can surpass human optimum in easy exploration game Pong. Environments such as robotics and real scenarios can be more complex than Atari. The reinforcement learning environment is commonly equipped with sparse reward, deceptive reward, confused state, complex state distribution, etc.; all these traps may bring oscillatory output or local optimum; however, most existing exploration methods just concentrate on one trap. We will focus on the combination of them in this article.

Existing tremendous exploration methods are spontaneously divided into two categories: endogenous exploration and exogenous exploration, in terms of the generation of exploration's action derived from intrinsic factors of the environment or not, such as states and goals.

Representative endogenous exploration focuses on the guidance "explore what surprises you" [1], such as curiosity-driven/intrinsic motivation/novelty-based methods [5–13].

State-based intrinsic exploration strategies use various indicators such as prediction error and information gain to describe the intrinsic reward signal. Exploration based on the variation in agent prediction error or learning progress is a typical method.

Following the strong representation ability of neural networks, state-of-the-art literature focuses on learning the intrinsic reward/exploration bonus from the state/state-action pairs, rather than applying the predefined indicators. References [5, 9] experiment on ignoring the extrinsic reward to keep away from getting stuck in the deceptive reward problems.

For most of the "real scenarios," this hypothesis is reasonable, and the intrinsic reward motivated by the final goal is actual existence; that is to say, the goal of the learning phase is observable. The computed intrinsic reward may not encourage the agent to explore in a high-return direction. However, for the environments whose goals/high reward states cannot be observed or inferred from state, these curiosity-driven methods may be powerless. A typical example in OpenAI Gym [14] is the benchmark Mountain Car environment; its states are continuous and actions are discrete. This game guides an underpowered car to reach goal on the top of a mountain. Horizontal position and velocity compose the state space, and their values are continuous. Legal actions are $\{-1, 0, 1\}$ which represent a scalar acceleration. The agent may be trapped into local optimization if the car does not reach the mountaintop goal as quickly as possible.

State of this environment has no indistinctive features to depict a goal, although the rendered frame shows a red flag to the programmer during training and testing procedures. Things will change if the state feed to the agent is the raw observation of the screenshot we can see; distance between the position of the car and the target flag can be extracted as the feature of the current state.

Other exploration algorithms, to a certain degree, can jump out of the trap. For endogenous exploration such as count-based methods, evolutionary computation techniques, and hindsight experience replay, taking the states of the whole trajectory/episode into consideration may be efficient. Exogenous exploration, which has no relationship to the inner model of the environment, may not face this dilemma.

Exogenous exploration suffers from more outlier effectiveness such as action perturbation [2, 3, 15, 16], Bayesian uncertainty estimation [17–20], parameter space noise [21], or specified reward [22].

Action perturbation exploration [23], alias of sophisticated/dithering exploration strategies, executes exploration process relying on dithering strategy, such as a random selection of the valid actions decided by a probability ϵ at the current step in the case of DQN [2, 3]. In the case of deep deterministic policy gradient (DDPG) [15, 16], the agent executes exploration by adding limited noise (maybe Gaussian noise or the more advanced Ornstein–Uhlenbeck noise) to action, which leads to an optimal state-action value at certain step. These strategies suffer inefficient performance in the case of RL problems with multidimensional continuous actions. Gaussian noises or OU noise may be suboptimal, and in practice, however, the hyperparameters which greatly affect the results are difficult to tune. Bayesian uncertainty estimation [17–20] utilized the bootstrap with random initialization, evaluated the uncertainty of neural networks with low computational cost, and made further improvement on deep exploration.

Exogenous exploration strategies act more universally, as they do not rely on the properties of the environments, while the endogenous exploration strategies need more specific design to adapt the environment to be confronted.

In this paper, we introduce a more general frame to make the best of both endogenous exploration and exogenous exploration, which encourage the agent to explore efficiently through the intrinsic reward signal produced by states and encourage the reward signal coming from the diverse goal imagination inspired by the goal exploration processes [23, 24] to interact with a trap group that may be encountered in environment.

2. Related Work

Evolutionary computation techniques, focusing on the exploration phase that can be archived to episode-based intrinsic exploration, have emerged as a convincing competitor of deep RL in the continuous action domain [23, 25–28].

Due to more attention to exploration, evolutionary computation techniques search policy directly in the policy parameter space, which results in a good performance in hard exploration situations such as rare reward environment or deceptive reward environment. Compare to SGD-based methods, the evolutionary computation techniques are generally less sample efficient as they lack gradient computations.

Pseudocount-based method, drawing inspiration from the intrinsic motivation literature, combined a mixed Monte Carlo update with a generated exploration bonus to achieve state-of-the-art on the notorious Montezuma's Revenge at that time [4]. The critical pseudocount was derived from an arbitrary density model, which is a

generated model to measure the uncertainty of the input state and is utilized by the pleasant theoretical guarantees of count-based exploration methods. Proof was given to demonstrate the close relationship between pseudocounts and information gain, which is widely applied to calculate novelty or curiosity. By introducing an information/prediction gain to measure the log-probability's delta value of two assignments, the authors consequently set information/prediction gain as intrinsic reward to perform count-based exploration.

Exploration with Exemplar (EX2) Models algorithm assessed how simple it is to discriminate between current state and states seen previously and evaluated states' novelty by the simplicity [29].

Reference [6] applied the misprediction error of a learned representation of states to estimate states' novelty. The agent was given exploration bonuses for visiting novel states. In this setting, the agent trained a dynamics model through the learned representation.

3. Preliminaries

Consider a Markov decision process (MDP), defined by the tuple (S, A, T, R, γ). S represents state spaces; A represents action spaces. T: $S \times A \times S \longrightarrow [0; 1]$ represents the transition distribution which is unknown in the reinforcement learning setting; reward function R: $S \times A \times S \longrightarrow \mathbb{R}$ is unknown, and the value at each time step can be queried through the agent-environment interaction; γ is the discount factor to control the importance of future versus immediate rewards.

In reinforcement learning, the agent learns to maximize the expected sum of discounted rewards, $\pi^* = \text{argmax}_\pi \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$, where τ denotes a trajectory $(s_0, a_0, \ldots, s_T, a_T)$ and π^* is the optimal policy.

Deep deterministic policy gradient (DDPG) is policy-based reinforcement learning. Unlike the value-based DQN agent which chooses action relying on value estimation, the DDPG agent's action is directly computed by a policy π, mapping states to a probability distribution over the actions π: $S \longrightarrow P(A)$. The action-value function $Q^\pi(s_t, a_t) = \mathbb{E}_\pi[R_t|s_t, a_t]$ depicts the expected return of (s_t, a_t) under policy π. The DDPG agent consists of an actor function $\mu(s|\theta^\mu)$ (acting as a policy) and a critic function $Q(s, a)$ (acting as a value estimator). Parameterized actor function $\mu(s|\theta^\mu)$ maps states to a specific action under the current parameterized policy and makes updates according to the chain rule with respect to the actor parameters.

$$\nabla\mu = \mathbb{E}_{\mu'}\left[\nabla_a Q\left(s, a|\theta^Q\right)|_{s=st, a=\mu(st)} \nabla\theta_\mu \mu\left(s|\theta^\mu\right)|_{s=st}. \right. \tag{1}$$

The critic function $Q(s, a)$ is updated according to Bellman equation as in deep Q-learning. Unlike widely used ϵ-greedy strategy, in this work exploration policy is defined as $\mu'(s_t) = \mu(s_t|\theta) + N$, adding noise sampled from a noise process N to the actor policy.

4. Count-Based Exploration and Episodic Memory

In low-dimensional Markov decision processes, table reinforcement learning incorporated within count-based exploration works well for states of the Markov decision processes [30] that can be easily exhausted. It is generally accepted that count-based exploration strategies turn inefficient when applied to high-dimensional Markov decision processes (generally high-dimensional state spaces, continuous action spaces, or both) since most states occur only once in deep reinforcement learning. Exploration methods widely applied in deep reinforcement learning rely on heuristic intrinsic motivation to explore unseen states or unreached parts of one state [30].

It is verified that the hippocampus together with the related internal temporal lobe structure in brain supports fast learning. The laboratory rat may be lost in navigation task due to its lesioned hippocampus or temporal lobe. Learning mechanism of the hippocampus is generally recognized as instance-based, while the cortex learns to generalize the representation of input distribution relatively.

The episodic memory module simulates the work of the hippocampus in the human brain. This is exactly the memory of past experience. It seems naturally and logically to apply episodic memory to count the situations encountered. Therefore, we use the contextual memory module to remember the states that the agent has encountered, as a count of states, and the purpose of exploration is to reduce the probability of encountering these states again. The purpose of exploration is to counter the episodic memory.

Inspired by model-free episodic control [31], we set the reward of the last state of one rollout as 1, $R_c(T) = 1$; when discount factor $\gamma = 1$, state value of each state experienced in the rollout can be $C(s_i) = 1$, $i = 1...T$; in other words, the count value of each state is 1, $C_{\text{count}}(s_i) = 1$. The C value is updated as follows:

$$C_{\text{count}}(s_t, a_t) \leftarrow \begin{cases} R_{ct}, & \text{if } (s_t, a_t) \notin C_{\text{count}}, \\ \max\{C_{count}(s_t, a_t), R_{ct}\}, & \text{otherwise.} \end{cases} \tag{2}$$

When encountering a state that has never been seen before, the C_t value is assigned to $R_c(t)$.

$$C_{\text{count}}(s, a) = \begin{cases} \frac{1}{k}\sum_{i=1}^{k} C_{\text{count}}\left(s^{(i)}, a\right), & \text{if } (s, a) \notin C_{\text{count}}, \\ C_{count}(s, a), & \text{otherwise.} \end{cases} \tag{3}$$

A critical process is to make decisions on when to explore and when to exploit; an indicator is designed to measure the exploration degree of current state, which is set to the ratio of delta between the maximum and the minimum counter; the agent explores when indicator is greater than the previously set threshold value; otherwise it exploits.

$$\frac{\max C_{count}(s_t) - \min C_{count}(s_t)}{\max C_{count}(s_t)} \leq \zeta. \tag{4}$$

We attach state counter to episodic memory, benefiting from the mechanism of episodic control algorithm. During the process of searching and updating, there is no need to establish another tree structure or to occupy other extra computing resources or memory.

5. Experiments

To verify in practice whether CounterEM learns more data efficiently, Atari Learning Environment [32, 33] which consists of various reward structures and exploration levels was chosen as a problem domain. We test our approach on Atari games that contain a series of interesting tasks such as sparse rewards and scores across different games. Pervious work had done a lot to apply the commonly used algorithms such as DQN and A3C and their variants in Atari Learning Environment and can be taken as baselines.

Reference [34] reproduces taxonomy of games in Atari Learning Environment on the basis of their exploration difficulty. Rough taxonomy of the games of Atari is "sparse" or "dense" rewards which depict the game's reward structure qualitatively. Limited by computing resources, we chose the seven notorious "sparse" rewards hard exploration games: Freeway, Gravitar, Montezuma's Revenge, Pitfall!, Private Eye, Solaris, and Venture; ten "dense" rewards hard exploration games: Alien, Amidar, Bank Heist, Frostbite, HERO, Ms. Pac-Man, Q ∗ bert, Surround, Wizard of Wor, and Zaxxon; and ten easy exploration games: Bowling, Breakout, Pong, Space Invaders, Boxing, Seaquest, Skiing, Demon Attack, Enduro, Gopher.

5.1. Experimental Parameters. For A3C, we run 100 rollout steps before being trained with 50 random batches of samples from the replay buffer. The cycle is repeated 20 times (2000 steps in the environment) before A3C is evaluated offline on 10000 steps (10 episodes). Replay buffer is a sliding window of size 10^6.

20 different seeds were used to reduce the variance of statistically different results. The inverse model first maps the input state (s_t) to a feature vector $\phi(s_t)$ using a series of two hidden layers of size (128, 128). For the inverse model, $\phi(s_t)$ and $\phi(s_{t+1})$ are concatenated into a single feature vector and passed as inputs into a fully connected layer of 64 units. The forward model is constructed by concatenating $\Phi(st)$ with at and passing them into a sequence of two fully connected layers with 64 and 128 units, respectively. The value of β is set to 0.2, while λ is set to 0.1. The batch size is set to 64, the discount factor is set to 0.99, and the actor and critic networks are designed with the same structure of two hidden layers of size (64, 64) with RELU activation functions. What is different is their output layer activation function; actor network output layer activation function is tanh while the critic network is linear.

The learning rates are 10−4 and 10−3, respectively, and Adam is used to optimize the loss function. The OU noise used in the A3C and the variant DQN algorithms linearly decreased from 0.9 at the first step to 0.1 at the final step. The performance was reported over 100 evaluation episodes of the best policy found during training process; each episode is set to 500 steps on the games.

5.2. Results. Table 1 summarizes the experimental results and data efficiency. CounterEM (NEC) and CounterEM (MFEC) significantly outperformed all other algorithms at small training step (less than 10 million frames). The gap is especially observed before 20 million frames (Algorithms 1 and 2).

Equipped with CounterEM, MFEC and NEC have a clear advantage in the initial learning stage, especially before 4 million frames. With the increase of training frames, CounterEM's efficiency gradually decreases. However, it is worth noting that CounterEM (NEC) outperformed the other baseline algorithms, training 40 million frames at its 10 million frames, which means more than 100 hours of training time.

In most of the Atari games, CounterEM (NEC) outperformed CounterEM (MFEC) on average, and both learned significantly faster in the initial phase than other baseline algorithms (see Table 2). At 2 million frames, CounterEM (MFEC) outperformed NEC. MFEC and NEC without CounterEM applied inefficient random exploration, which becomes even less efficient as the number of actions increases. Thanks to proven count-based exploration methods, CounterEM directs agents to explore unseen or rare-seen states to obtain high rewards quickly. However, when the training step increases up to a certain threshold, which may be positively correlated to the dimensions of states and actions, the superiority of CounterEM may weaken.

It is worth noting that, in order to ensure the stability of training, the baseline algorithms A3C and DQN and related variant algorithms need to crop the reward to the range of [−1, 1] [2, 3]. NEC and MFEC do not need reward clips, and therefore CounterEM (NEC) and CounterEM (MFEC) do not require reward clips. This resulted in quality changes in behavior and better performance than other games that required editing (such as Alien, Frostbite, Pac-Man, Bowling, and HERO). The counter estimator is naturally set to the [−1, 1] range, but this does not affect the agent's learning efficiency in these games because we do not use the counter estimator directly in the Q calculation.

5.3. Experiment for Object Detection. To test our module in high dimensions, we turn to object localization/object detection which plays an important role in the computer vision field. RL-based target detection and target tracking usually use the standard A3C algorithm, and they pay more attention to the design of the framework and network structure, while action disturbance is used for exploration. As a very important link in RL, exploration also plays an important role in this application field. Its goal is to place bounding boxes in a given image around the instances of predefined object class, such as faces, ships, and desktops. During localization process, detectors analyze the scanning windows of the input image, while the transformation of windows is guided by scales and locations. Most state-of-the-art solutions for object detection are bottom-up region

Table 1: Median across games of human-normalized scores for listed algorithms at different training frames.

Frames (M)	Nature DQN (%)	Retrace(λ) (%)	A3C (%)	MFEC (%)	CounterEM (MFEC) (%)	NEC (%)	CounterEM (NEC) (%)
1	−0.7	−0.4	0.4	12.8	20.1	16.7	29.6
2	0.0	0.2	0.9	16.7	29.1	27.8	37.6
4	2.4	3.3	1.9	26.6	36.2	36.0	48.4
10	15.7	17.3	3.6	45.4	52.5	54.6	69.3
20	26.8	30.4	7.9	55.9	66.1	72.0	77.6
40	36.7	60.5	18.4	61.9	70.0	83.3	84.5

(1) **for** episode = 1 to S **do**
(2) **for** $t = 1$ to T **do**
(3) Obtain observation s_t from the environment
(4) Let $s_t = \phi(o_t)$
(5) Estimate C_{count} and Q for each action a via (3)
(6) **if** Satisfy (4) **then**
(7) Choose $a_t = \text{argmin}_a C_{\text{count}}(s_t, a_t)$
(8) **else**
(9) Choose $a_t = \text{argmax}_a Q^{EC}(s_t, a)$
(10) **end if**
(11) Execute action a_t, and receive reward $r_t + 1$
(12) **end for**
(13) **for** $t = T$ to 1 **do**
(14) Update $Q^{EC}(s_t, a_t)$ and $\widehat{C_{\text{count}}}(s, a)$ according to (2)
(15) **end for**
(16) **end for**

Algorithm 1: Exploration for countering model-free episodic control.

(1) Initialize replay memory D
(2) Initialize a DND M_a for each action a
(3) Initialize N for horizon of the N-step Q rule
(4) **for** episode = 1 to S **do**
(5) **for** $t = 1$ to T **do**
(6) Obtain observation s_t from the environment with embedding
(7) Estimate $Q(s_t, a)$ for each action a via (2) from M_a
(8) **if** Satisfy (4) **then**
(9) Choose $a_t = \text{argmin}_a C_{\text{count}}(s_t, a_t)$
(10) **else**
(11) Choose $a_t = \text{argmax}_a Q^{EC}(s_t, a)$
(12) **end if**
(13) Execute action a_t, and receive reward $r_t + 1$
(14) Append $(h, Q^{(N)}(s_t, a_t), C^{(N)}_{\text{count}}(s_t, a_t))$ to M_{at}
(15) Append $(s_t, a_t, Q^{(N)}(s_t, a_t), C^{(N)}_{\text{count}}(s_t, a_t))$ to D
(16) Train a random minibatch in D
(17) **end for**
(18) **end for**

Algorithm 2: Exploration for countering neural episodic control.

Table 2: Mean human-normalized scores for listed algorithms at different training frames.

Frames (M)	Nature DQN (%)	Retrace(λ) (%)	A3C (%)	MFEC (%)	CounterEM (MFEC) (%)	NEC (%)	CounterEM (NEC) (%)
1	−10.5	−10.5	5.2	28.4	40.8	45.6	61.7
2	−5.8	−5.4	8.0	39.4	68.3	58.3	84.9
4	8.8	6.2	11.8	53.4	88.4	73.3	97.3
10	51.3	52.7	22.3	85.0	100.1	99.8	115.4
20	94.5	237.7	59.7	113.6	117.9	121.5	122.8
40	151.2	386.5	255.4	142.2	148.1	144.8	150.0

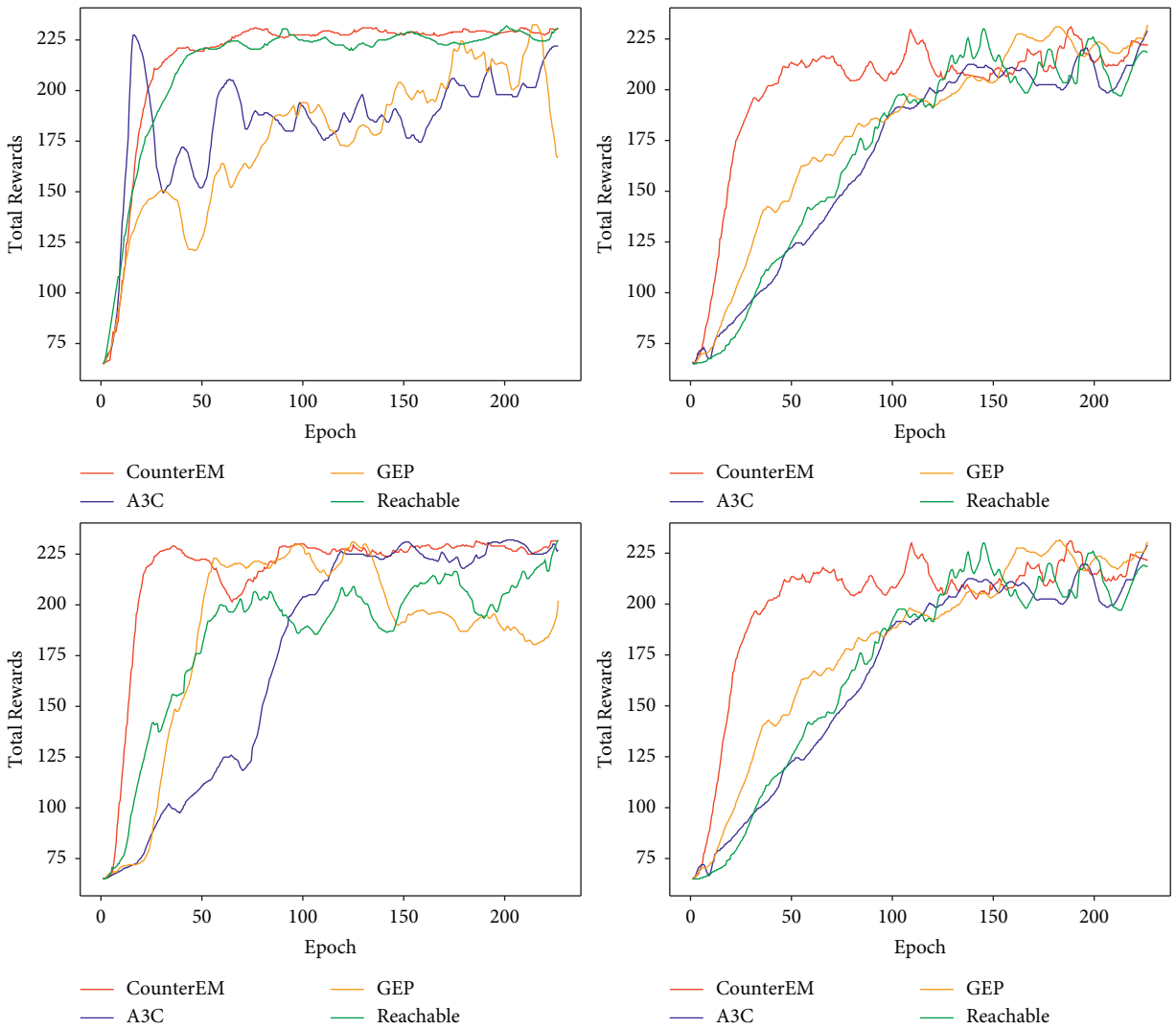

Figure 1: Sum reward in computer vision applications.

proposals [35, 36]; thousands of windows were selected and evaluated one by one. These bottom-up methods were accelerated by the advancement of convolutional neural networks (CNNs) and parallel computing benefits from rapid expansion of graphic processing unit (GPU) [37–40].

Current active search methods, reformulated for learning a navigation strategy, based on the DQN frame, artificially designed several actions (horizontal/vertical moves in fixed pixels, scale changes in fixed scale related to the pixels, aspect ratio changes in fixed ratio related to the pixels, trigger) to form an action set [41–47]. Agent selected action (a) that generates the max estimated action value (Q); that is to say, the policy $\pi = P(a|s)$ would not exist independently of the action-value estimation, which leadd to continuous action space situation beyond the off-policy algorithm as maximum action value is not easy to figure out.

5.3.1. State. The state of the MDP consists of the whole image feature vector, current time step window feature vector, and history of performed actions in current episode. These three elements are simply concatenated into a new vector to represent the state. The features are extracted using a pretrained VGG-16 model [48] for both the whole image and current window. Feature vector of layer fc6 $r_t + 1$ was applied in our experiments, and the VGG-16 was pretrained on ImageNet.

5.3.2. Action. An action space $A(s)$ defines the legal action in any given state $s \in S$; at each time step, the agent performs action to deploy the box which surrounds the object. In the 2D object detection application, four possible actions, up, down, left, and right, allow for pixel-wise movement being universal solution.

5.3.3. Reward Function. The agent receives a new visual observation of the environment s_{t+1} and a reward signal r_{t+1} when performing the action obtained from the agent.

The agent makes decision to maximize the sum of the reward signal $R = \sum_i r_i$, while in application fields, it is usually very sparse and hysteric.

To simulate the common situation, in our object detection experiment, we set $r_T = 1$ only when object is classified correctly and 0 otherwise.

5.3.4. Results. Figure 1 is an experimental comparison diagram of several applications in the CV field. We use the three different exploration methods of OU noise [15, 16], GEP [23], and Reachable [34, 49–53] for comparison with our CounterEM method, in pedestrian tracking and face detection. Experiments show that CounterEM can get good rewards quickly.

6. Conclusions

Episodic memory can be used for episodic control and can achieve good results in some RL application situations. In the case of relatively easy exploration or nonsparse reward, the agent can find the path of high reward very smoothly, even when reward is sparse. The problems of reward and hard exploration are dependent more on the strategy of exploration. In this article, we modified the episodic memory module to pseudocount state, so as to realize a pseudocount-based exploration strategy. The experiment shows that our algorithm can achieve good results in OpenAI games as well as computer vision applications such as object detection and object tracking. In the next step, we plan to expand the counter episodic memory to continuous episodic control. Dual networks seem to be a feasible solution, because their inputs are all the states at certain time, and their embeddings are consistent.

Data Availability

The game data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. S. Sutton and A. G. Barto, "Reinforcement learning: an introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, p. 1054, 1998.

[2] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2017.

[3] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous Methods for Deep Reinforcement Learning," 2016.

[4] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying Count-Based Exploration and Intrinsic Motivation," vol. 1606, 2016, https://www.researchgate.net/publication/303822096_Unifying_Count-Based_Exploration_and_Intrinsic_Motivation.

[5] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," 2017, https://www.researchgate.net/publication/316955874_Curiosity-driven_Exploration_by_Self-supervised_Prediction.

[6] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models," 2015, https://www.researchgate.net/publication/279808581_Incentivizing_Exploration_In_Reinforcement_Learning_With_Deep_Predictive_Models.

[7] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," 2018.

[8] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical Deep Reinforcement Learning - Integrating Temporal Abstraction and Intrinsic Motivation," 2016.

[9] M. C. Machado and M. H. Bowling, "Learning Purposeful Behaviour in the Absence of Rewards," 2016.

[10] N. Shaker, "Intrinsically motivated reinforcement learning: a promising framework for procedural content generation," in *Proceedings of the 2016 IEEE Conference on Computational Intelligence and Games (CIG)*, IEEE, Santorini, Greece, 2017.

[11] H. He and X. Zhong, "Learning without external reward [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 48–54, 2018.

[12] M. Andrychowicz, D. Crow, A. Ray et al., "Hindsight Experience Replay," 2017.

[13] C. Finn, S. Levine, and P. Abbeel, "Guided Cost Learning - Deep Inverse Optimal Controlvia Policy Optimization," 2016, https://www.researchgate.net/publication/301872691_Guided_Cost_Learning_Deep_Inverse_Optimal_Control_via_Policy_Optimization.

[14] G. Brockman, V. Cheung, L. Pettersson et al., "OpenAI Gym," 2016.

[15] T. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," 2015, https://www.researchgate.net/publication/281670459_Continuous_control_with_deep_reinforcement_learning.

[16] A. Pritzel, B. Uria, and S. Srinivasan, "Neural episodic control," in *Proceedings of the International Conference on Machine Learning*, pp. 2827–2836, T. Haoran, H. Rein, and F Davis, Sydney, Australia, 2017.

[17] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep Exploration via Bootstrapped DQN," 2016, https://www.semanticscholar.org/paper/Deep-Exploration-via-Bootstrapped-DQN-Osband-Blundell.

[18] I. Osband, J. Aslanides, and A. Cassirer, "Randomized prior functions for deep reinforcement learning," 2018, https://www.researchgate.net/publication/325709368_Randomized_Prior_Functions_for_Deep_Reinforcement_Learning.

[19] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: representing model uncertainty in deep learning," vol. 1506, 2015, https://www.researchgate.net/publication/277959098_Dropout_as_a_Bayesian_Approximation_Representing_Model_Uncertainty_in_Deep_Learning.

[20] R. Houthooft, X. C. 0022, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "VIME - Variational Information Maximizing Exploration," in *Proceedings of the Neural Information Processing Systems (NIPS)*, Barcelona, Spain, 2016.

[21] M. Plappert, R. Houthooft, P. Dhariwal et al., "Parameter Space Noise for Exploration," 2017, https://www.researchgate.net/publication/317399634_Parameter_Space_Noise_for_Exploration.

[22] T. Xu, Q. Liu, L. Zhao, and J. Peng, "Learning to Explore via Meta-Policy Gradient," 2018, https://www.researchgate.net/publication/323770820_Learning_to_Explore_with_Meta-Policy_Gradient.

[23] C. Colas, O. Sigaud, and P.-Y. Oudeyer, "GEP-PG - Decoupling Exploration and Exploitation in Deep Reinforcement Learning Algorithms," 2018.

[24] S. Forestier, Y. Mollard, and P.-Y. Oudeyer, "Intrinsically Motivated Goal Exploration Processes with Automatic

Curriculum Learning," 2017, https://www.researchgate.net/publication/318981596_Intrinsically_Motivated_Goal_Exploration_Processes_with_Automatic_Curriculum_Learning.

[25] H. Liu, Y. Pan, S. Li, and Y. Chen, "Synchronization for fractional-order neural networks with full/under-actuation using fractional-order sliding mode control," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 7, pp. 1219–1232, 2018.

[26] H. Liu, S. Li, G. Li, and H. Wang, "Robust adaptive control for fractional-order financial chaotic systems with system uncertainties and external disturbances," *Information Technology and Control*, vol. 46, no. 2, pp. 246–259, 2017.

[27] T. Salimans, J. Ho, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, https://www.researchgate.net/publication/314943017_Evolution_Strategies_as_a_Scalable_Alternative_to_Reinforcement_Learning.

[28] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," 2016.

[29] J. Fu, J. D. Co-Reyes, and S. Levine, "EX2 - Exploration with Exemplar Models for Deep Reinforcement Learning," vol. 1703, 2017, https://www.researchgate.net/publication/314237647_EX2_Exploration_with_Exemplar_Models_for_Deep_Reinforcement_Learning.

[30] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, and Y. Duan, "Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning," 2016, https://www.researchgate.net/publication/310329157_Exploration_A_Study_of_Count-Based_Exploration_for_Deep_Reinforcement_Learning.

[31] C. Blundell, U. Benigno, and P. Alexander, "Model-free episodic control," 2016, https://www.researchgate.net/publication/303970053_Model-Free_Episodic_Control.

[32] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: an evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[33] M. G. Bellemare, J. Veness, and E. Talvitie, "Skip Context Tree Switching-Supp," 2014, https://www.semanticscholar.org/paper/Skip-Context-Tree-Switching-Bellemare-Veness.

[34] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos, "Count-based exploration with neural density models," in *Proceedings of the International Conference on Machine Learning*, pp. 2721–2730, PMLR, Sydney, Australia, 2017.

[35] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[36] R. Girshick, J. Donahue, and T. Darrell, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, Ohio, OH, USA, 2014.

[37] R. Girshick, "FastR-CNN." in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, IEEE Computer Society, 2015.

[38] S. Ren, K. He, R. Girshick, and S. Jian, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 91–99, MIT Press, Montréal, Canada, 2015.

[39] R. Girshick, J. Donahue, T. Darrell, and M. Jitendra, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, IEEE Computer Society, Columbus, OH, USA, 2014.

[40] K. He, X. Zhang, S. Ren, and S. Jian, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2014.

[41] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," in *Proceedings of the IEEE International Conference on Computer Vision. IEEE*, pp. 2488–2496, 2015.

[42] M. Bellver, X. Giro-I-Nieto, F. Marques, and T. Jordi, "Hierarchical Object Detection with Deep Reinforcement Learning," *Advances in Parallel Computing*, vol. 31, no. 164, 2016.

[43] X. Kong, B. Xin, Y. Wang, and H. Gang, "Collaborative Deep Reinforcement Learning for Joint Object Search," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7072–7081, Honolulu, Hawaii, 2017.

[44] F. C. Ghesu, B. Georgescu, and Y. Zheng, "Multi-scale deep reinforcement learning for real- time 3D-landmark detection in CT scans," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, pp. 176–189, 2017.

[45] S. Mathe, A. Pirinen, and C. Sminchisescu, "Reinforcement learning for visual object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2894–2902, Las Vegas, NV, USA, 2016.

[46] V. Mnih, N. Heess, and A. Graves, "Recurrent models of visual attention," *Advances in Neural Information Processing Systems*, pp. 2204–2212, 2014.

[47] Z. Jie, X. Liang, and J. Feng, "Tree-structured reinforcement learning for sequential object localization," *Advances in Neural Information Processing Systems*, vol. 29, pp. 127–135, 2016.

[48] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://www.researchgate.net/publication/265385906_Very_Deep_Convolutional_Networks_for_Large-Scale_Image_Recognition.

[49] N. Savinov, A. Raichuk, and D. Vincent, "Episodic curiosity through reachability," 2019, https://www.researchgate.net/publication/328091546_Episodic_Curiosity_through_Reachability.

[50] H. Hu, "Generalizable Episodic Memory for Deep Reinforcement Learning," 2021, https://www.researchgate.net/publication/350005462_Generalizable_Episodic_Memory_for_Deep_Reinforcement_Learning.

[51] S. Jabbari, M. Joseph, M. J. Kearns, J. Morgenstern, and A. Roth, "Fairness in Reinforcement Learning," 2017.

[52] J. C. Nunnally, "Explorations of exploration," *Advances in Intrinsic Motivation and Aesthetics*, pp. 87–129, 1981.

[53] T. Barron, O. Obst, and H. Ben Amor, "Information Maximizing Exploration with a Latent Dynamics Model," 2018, https://www.semanticscholar.org/paper/Information-Maximizing-Exploration-with-a-Latent-Barron-Obst.